[0057] CLAIMS

What is claimed is:

1. A method comprising simulating the execution of all execution paths of one or more assemblies in managed code to find a set of required permissions for each said execution path, wherein:

the managed code is a managed shared library or an executable; and

each call in each execution path has a corresponding said permissions set.

2. The method as defined in Claim 1, wherein the execution paths for only one said assembly in managed code are simulated to find the set of required permissions for each said execution path by a union of the permissions for each said execution path.

3. The method as defined in Claim 1, wherein:

the one or more assemblies in managed code correspond to an application; and

the set of required permissions for each said execution path comprises a union of the permissions for each said execution path.

4. The method as defined in Claim 1, wherein:

the assemblies in managed code correspond to a shared library; and

the set of required permissions for each said execution path comprises one separate permission set per entry point in the shared library.

5. The method as defined in Claim 1, wherein the set of required permissions for each said execution path comprises a union of the permissions for each said execution path.

6. The method as defined in Claim 1, wherein one of more of the calls in at least one said execution path is an cross assembly call.

7. The method as defined in Claim 1, wherein:

the managed code is built to make use of a common language runtime;

each said assembly is packaged as an executable entity or as a data link library entity and

each said assembly includes one or more methods.

8. The method as defined in Claim 7, wherein the simulation of the execution of each said execution path comprises a simulation of the flow of argument data using intra and extra method data flow analysis for each said method.

9. The method as defined in Claim 1, wherein when the executable has permissions to execute that are not less than a union of permission sets for each said execution path, any dynamic execution of the executable will not trigger a security exception.

10. The method as defined in Claim 1, wherein the simulation of the execution comprises, for each said execution path, one or more simulated stack walks that each include a plurality of said assemblies.

11. A computer readable medium including machine readable instructions for implementing the method as defined in claim 1.

12. In a managed code environment, a method comprising:

simulating calling from one assembly to another for which a permission set is required, each said assembly being managed code in a library or corresponding to an application;

repeating the calling for each said assembly in the managed code and for all possible execution paths of the managed code; and

finding the union of the permissions sets corresponding to each said call.

13. The method as defined in Claim 12, wherein the manage code environment comprises:

a managed code portion including:

the assemblies; and

a virtual machine;

a native code portion including:

an execution engine for the virtual machine; and

an operating system under the execution engine.

14. The method as defined in Claim 12, wherein:

the managed code is built to make use of a common language runtime;

each said assembly is packaged as an executable entity or as a data link library entity and

each said assembly includes one or more methods.

15. The method as defined in Claim 12, wherein when the assemblies corresponding to the application have permissions to execute that are not less than the union of permission sets for each said execution path, any dynamic execution of the assemblies corresponding to the application will not trigger a security exception.

16. The method as defined in Claim 12, wherein the simulation of the execution comprises, for each said execution path, one or more simulated stack walks that include two or more of said assemblies.

17. The method as defined in Claim 12, wherein the managed code environment enforces partial trust security contexts.

18. A computer readable medium including machine readable instructions for implementing the method as defined in claim 12.

19. One or more computer-readable media comprising instructions that, when executed, perform a simulation of the execution of every data and control flow for managed code from which an estimate is derived of the minimum security requirements needed to dynamically execute the managed code without triggering a security exception.

20. The one or more computer-readable media as defined in Claim 19, wherein:

the managed code, which comprises a plurality of assemblies, is built to make use of a common language runtime;

each said assembly is packaged as an executable entity or as a data link library entity and

each said assembly includes one or more methods.

21. The one or more computer-readable media as defined in Claim 19, wherein the dynamic execution of the managed code occurs in a managed code environment comprising:

a managed code portion including:

the managed code has one or more assemblies and is a library or an executable; and

a virtual machine;

a native code portion including:

an execution engine for the virtual machine; and

an operating system under the execution engine.

22. The one or more computer-readable media as defined in Claim 21, wherein:

the managed code is built to make use of a common language runtime;

each said assembly is packaged as an executable entity or as a data link library entity and

each said assembly includes one or more methods.

23. The one or more computer-readable media as defined in Claim 21, wherein the simulation of the execution comprises, for each said data and control flow for the managed code, one or more simulated stack walks that include two or more of said assemblies.

24. The one or more computer-readable media as defined in Claim 21, wherein:

each call in each said simulated stack walk has a corresponding permissions set; and

the derived estimate is a union of the permissions sets.

25. The one or more computer-readable media as defined in Claim 21, wherein the managed code environment enforces partial trust security contexts.

26. An apparatus comprising:

virtual machine means, in a managed code portion, for operating a plurality of assemblies in managed code, wherein the managed code is a managed shared library or an executable and is in the managed code portion;

execution engine means, in a native code portion, for the virtual machine means;

means, in a native code portion, for providing an operating system;

means for making a call for access by one said assembly to another said assembly for which a permissions set is required;

means for gathering the permissions set from each said call;

means for deriving a union of the gathered permissions sets; and

means for simulating the execution of all possible execution paths for the managed shared library or the executable to derive therefrom the derived union of the gathered permissions sets.

27. The apparatus as defined in Claim 26, further comprising:

means for compiling the assemblies from an intermediate language code and metadata into native code; and

means for loading the native code with a Common Language Runtime (CLR) loader in the native code portion to load the compiled native code, wherein the execution engine means executes the compiled native code in the native code portion.

28. The apparatus as defined in Claim 26, wherein the managed code portion further comprises one or more files associated with user code that, when compiled into an intermediate language code and metadata generated by a language compiler, are represented by the assemblies.

29. The apparatus as defined in Claim 26, wherein the execution engine means in the native code portion further comprises a compiler to compile each said assembly into native code for execution by the native code portion.

30. The apparatus as defined in Claim 26, wherein the execution engine means in the native code portion further comprises:

a Just In Time (JIT) compiler to compile each said assembly into native code; and

a CLR loader to load the compiled native code for execution by the native code portion.

31. The apparatus as defined in Claim 26, further comprising:

means, in the native code portion, for forming a response to the call; and

means for returning the response to the first assembly in the managed code portion.

32. The apparatus as defined in Claim 26, wherein:

the managed code is built to make use of a common language runtime;

each said assembly is packaged as an executable entity or as a data link library entity; and

each said assembly includes one or more methods.

33. The apparatus as defined in Claim 32, wherein the simulation of the execution comprises, for each said execution path, a simulation of the flow of argument data using intra and extra data flow analysis for each said method.

34. The apparatus as defined in Claim 26, wherein when the executable has permissions to execute that are not less than the union of the gathered permissions sets, any dynamic execution of the executable will not trigger a security exception.

35. The apparatus as defined in Claim 26, wherein the simulation of the execution comprises, for each said execution path, one or more simulated stack walks that each include a plurality of said assemblies.

36. The apparatus as defined in Claim 35, wherein each call in each said simulated stack walk has a corresponding said permissions set.

37. The apparatus as defined in Claim 26, wherein the managed code portion and the native code portion are in a managed code environment that enforces partial trust security contexts.

38. A computing device comprising:

a managed code portion including:

a plurality of assemblies each being managed code in a managed shared library or in an executable; and

a virtual machine;

a native code portion including:

an execution engine for the virtual machine; and

an operating system under the execution engine;

logic configured to:

simulate the execution of all possible calls from one said assembly to another for all possible execution paths of the managed code, wherein each assembly call has a corresponding permissions set; and

derive a union of the permissions sets from each said assembly call.

39. The computing device as defined in Claim 38, wherein the managed code portion further comprises one or more files associated with user code that, when compiled into an intermediate language code and metadata generated by a language compiler, are represented by:

the assemblies in the executables; or

the managed shared library.

40. The computing device as defined in Claim 38, wherein the execution engine further comprises:

a compiler to compile each said assembly into native code; and

a CLR loader to load the compiled native code.

41. The computing device as defined in Claim 38, wherein:

the managed code is built to make use of a common language runtime;

each said assembly is packaged as an executable entity or as a data link library entity and

each said assembly includes one or more methods.

42. The computing device as defined in Claim 41, wherein the simulation of the execution comprises a simulation of the flow of argument data using intra and extra method data flow analysis for each said method.

43. The computing device as defined in Claim 38, wherein when the executable has permissions to execute that are not less than the union of the permissions sets from each said assembly call, any dynamic execution of the executable will not trigger a security exception.

44. The computing device as defined in Claim 38, wherein the simulation of the execution comprises one or more simulated stack walks that each include a plurality of said assemblies.

45. The computing device as defined in Claim 38, wherein the managed code portion and the native code portion are in a managed code environment that enforces partial trust security contexts.